

# P&S Processing-in-Memory

Real-World Processing-in-Memory Architectures:  
Samsung HBM-PIM Architecture

Dr. Juan Gómez Luna

Prof. Onur Mutlu

ETH Zürich

Fall 2022

8 November 2022

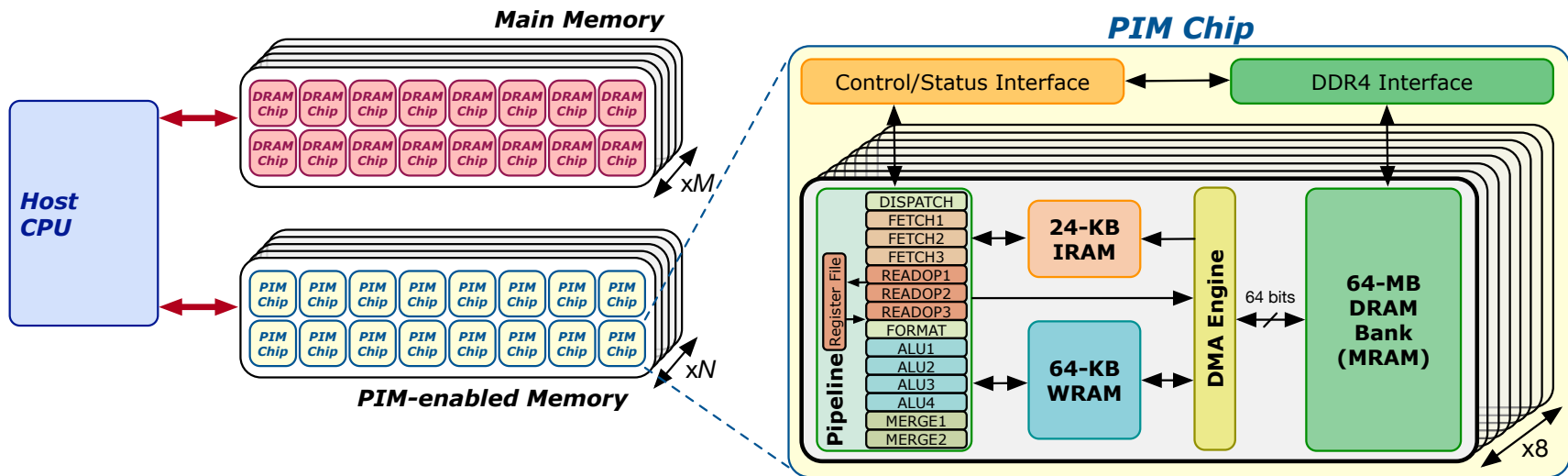
# UPMEM Processing-in-DRAM Engine (2019)

- **Processing in DRAM Engine**
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.
- Replaces **standard DIMMs**
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth



# Recall: UPMEM PIM System Organization

- A UPMEM DIMM contains 8 or 16 chips
  - Thus, 1 or 2 ranks of 8 chips each
- Inside each PIM chip there are:
  - 8 64MB banks per chip: Main RAM (MRAM) banks
  - 8 DRAM Processing Units (DPUs) in each chip, 64 DPUs per rank



# Experimental Analysis of the UPMEM PIM Engine

---

## Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland

IZZAT EL HAJJ, American University of Beirut, Lebanon

IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain

CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland

ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is insufficient to amortize the cost of main memory access. Fundamentally addressing this *data movement bottleneck* requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as *processing-in-memory* (PIM).

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPMEM company has designed and manufactured the first publicly-available real-world PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called *DRAM Processing Units* (DPUs), integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present *PrIM* (*Processing-In-Memory benchmarks*), a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,556 DPUs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.



# Understanding a Modern PIM Architecture



The video player shows a lecture titled "Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization". The speaker is Juan Gómez Luna, with co-authors Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu. The video includes links to the arXiv paper and GitHub repository. The player interface shows a progress bar at 2:26 / 2:57:10, and the channel name "Onur Mutlu Lectures" with 18.7K subscribers. The video has 2,579 views and was streamed live on Jul 12, 2021.

**Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization**

Juan Gómez Luna, Izzat El Hajj,  
Ivan Fernandez, Christina Giannoula,  
Geraldo F. Oliveira, Onur Mutlu

<https://arxiv.org/pdf/2105.03814.pdf>  
<https://github.com/CMU-SAFARI/prim-benchmarks>

ETH Zürich SAFARI

2:26 / 2:57:10

SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture

2,579 views • Streamed live on Jul 12, 2021

93 0 SHARE + SAVE ...

**Onur Mutlu Lectures**  
18.7K subscribers

SUBSCRIBED

# Samsung HBM-PIM, a.k.a. FIMDRAM

# Samsung Function-in-Memory DRAM (2021)



## Samsung Develops Industry's First High Bandwidth Memory with AI Processing Power

Korea on February 17, 2021

Audio



Share



*The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%*

Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry's first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power – the HBM-PIM. The new processing-in-memory (PIM) architecture brings powerful AI computing capabilities inside high-performance memory, to accelerate large-scale processing in data centers, high performance computing (HPC) systems and AI-enabled mobile applications.

Kwangil Park, senior vice president of Memory Product Planning at Samsung Electronics stated, "Our groundbreaking HBM-PIM is the industry's first programmable PIM solution tailored for diverse AI-driven workloads such as HPC, training and inference. We plan to build upon this breakthrough by further collaborating with AI solution providers for even more advanced PIM-powered applications."

# Function-in-Memory DRAM (ISSCC 2021)

---

## ISSCC 2021 / SESSION 25 / DRAM / 25.4

---

### **25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications**

Young-Cheon Kwon<sup>1</sup>, Suk Han Lee<sup>1</sup>, Jaehoon Lee<sup>1</sup>, Sang-Hyuk Kwon<sup>1</sup>,  
Je Min Ryu<sup>1</sup>, Jong-Pil Son<sup>1</sup>, Seongil O<sup>1</sup>, Hak-Soo Yu<sup>1</sup>, Haesuk Lee<sup>1</sup>,  
Soo Young Kim<sup>1</sup>, Youngmin Cho<sup>1</sup>, Jin Guk Kim<sup>1</sup>, Jongyoon Choi<sup>1</sup>,  
Hyun-Sung Shin<sup>1</sup>, Jin Kim<sup>1</sup>, BengSeng Phuah<sup>1</sup>, HyoungMin Kim<sup>1</sup>,  
Myeong Jun Song<sup>1</sup>, Ahn Choi<sup>1</sup>, Daeho Kim<sup>1</sup>, SooYoung Kim<sup>1</sup>, Eun-Bong Kim<sup>1</sup>,  
David Wang<sup>2</sup>, Shinhaeng Kang<sup>1</sup>, Yuhwan Ro<sup>3</sup>, Seungwoo Seo<sup>3</sup>, JoonHo Song<sup>3</sup>,  
Jaeyoun Youn<sup>1</sup>, Kyomin Sohn<sup>1</sup>, Nam Sung Kim<sup>1</sup>

<sup>1</sup>Samsung Electronics, Hwaseong, Korea

<sup>2</sup>Samsung Electronics, San Jose, CA

<sup>3</sup>Samsung Electronics, Suwon, Korea

## Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology

Industrial Product

Sukhan Lee<sup>§1</sup>, Shin-haeng Kang<sup>§1</sup>, Jaehoon Lee<sup>1</sup>, Hyeonsu Kim<sup>2</sup>, Eojin Lee<sup>1</sup>, Seungwoo Seo<sup>2</sup>,  
Hosang Yoon<sup>2</sup>, Seungwon Lee<sup>2</sup>, Kyoungwan Lim<sup>1</sup>, Hyunsung Shin<sup>1</sup>, Jinhyun Kim<sup>1</sup>,  
Seongil O<sup>1</sup>, Anand Iyer<sup>3</sup>, David Wang<sup>3</sup>, Kyomin Sohn<sup>1</sup> and Nam Sung Kim<sup>§1</sup>

<sup>1</sup>Memory Business Division, Samsung Electronics

<sup>2</sup>Samsung Advanced Institute of Technology, Samsung Electronics

<sup>3</sup>Device Solutions America, Samsung Electronics

## **Aquabolt-XL: Samsung HBM2-PIM with in-memory processing for ML accelerators and beyond**

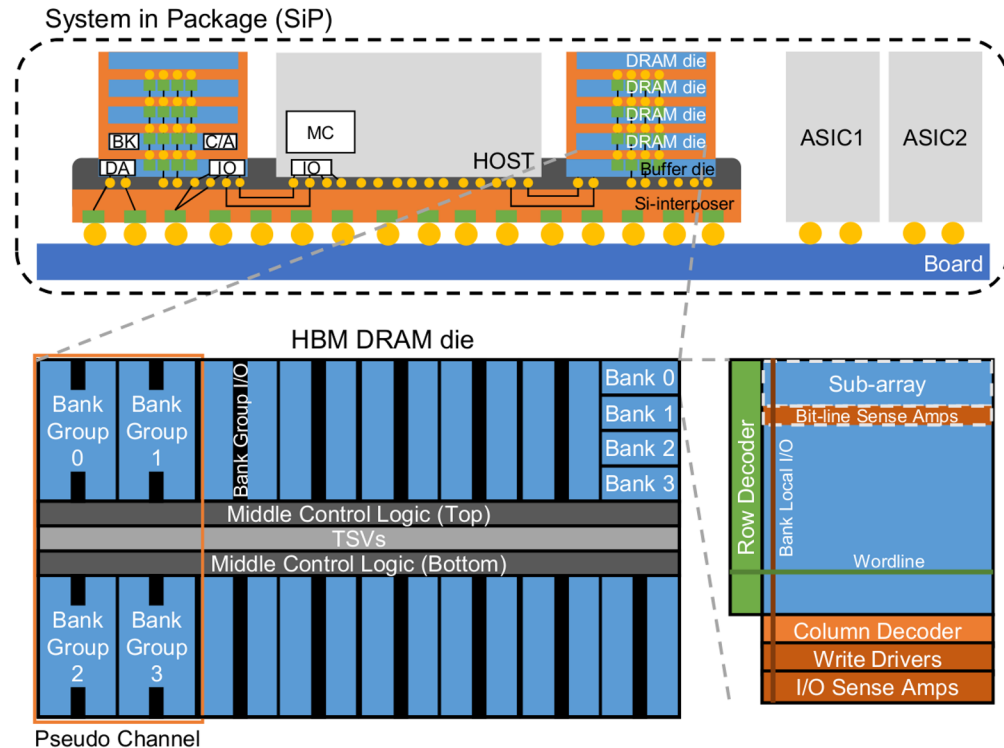
Jin Hyun Kim, Shin-haeng Kang, Sukhan Lee, Hyeonsu Kim, Woongjae Song, Yuhwan Ro, Seungwon Lee, David Wang, Hyunsung Shin, Bengseng Phuah, Jihyun Choi, Jinin So, YeonGon Cho, JoonHo Song, Jangseok Choi, Jeonghyeon Cho, Kyomin Sohn, Youngsoo Sohn, Kwangil Park, and Nam Sung Kim

Samsung Electronics



# Background: High Bandwidth Memory (HBM)

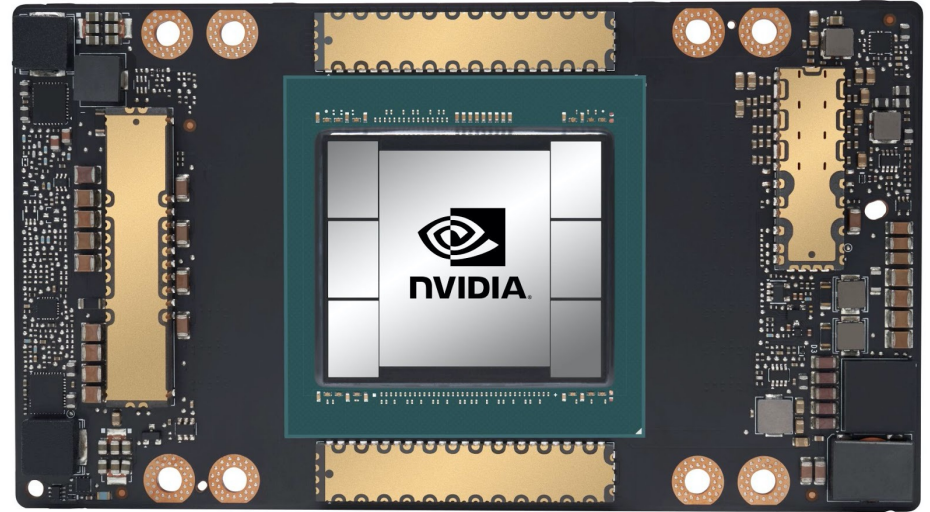
- HBM stacks **DRAM layers** and a **buffer layer**
  - The buffer layer contains I/O circuitry, self-test, test/debug
- DRAM layers and buffer layer communicate using **Through Silicon Vias (TSVs)**
- The buffer layer is connected to a host processor via a **silicon interposer**
- 1 HBM2 die comprises 4 pseudo channels (pCHs) each with 4 bank groups
  - An access transfers a 256-bit data block over 4 64-bit bursts over one pCH



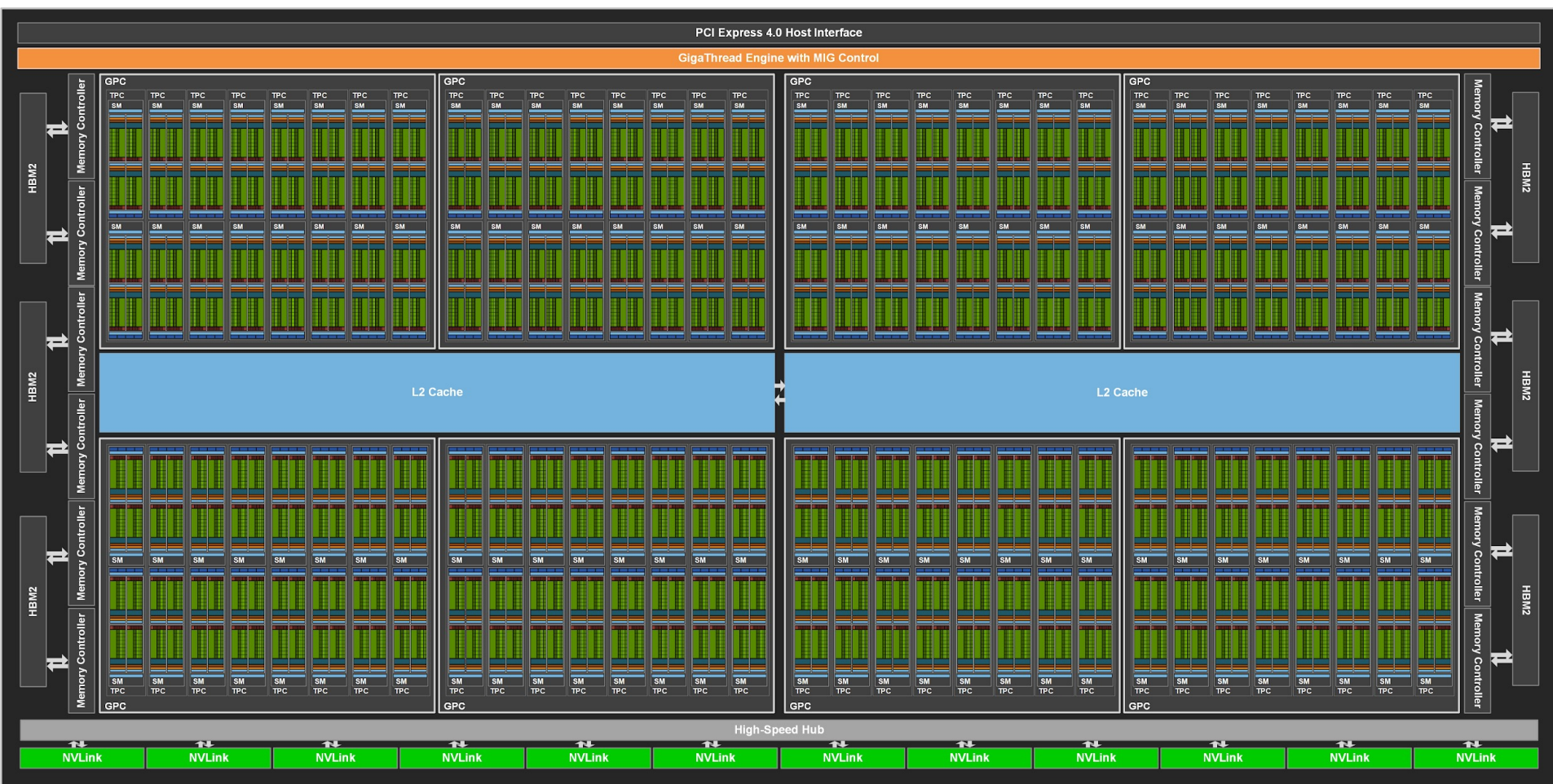
# NVIDIA A100 GPU

---

- NVIDIA-speak:
  - ❑ 6912 stream processors
  - ❑ “SIMT execution”
- Generic speak:
  - ❑ 108 cores
  - ❑ 64 SIMD functional units per core
  - ❑ Tensor cores for Machine Learning
    - Support for sparsity
    - New floating point data type (TF32)



# NVIDIA A100 Block Diagram



<https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/>

108 cores on the A100  
(Up to 128 cores in the full-blown chip)

40MB L2 cache

# NVIDIA H100 GPU

---

- NVIDIA-speak:
  - ❑ 14592 stream processors
  - ❑ “SIMT execution”
- Generic speak:
  - ❑ 144 cores
  - ❑ 64 SIMD functional units per core
  - ❑ Tensor cores for Machine Learning
    - New 8-bit floating point formats





# NVIDIA H100 Block Diagram



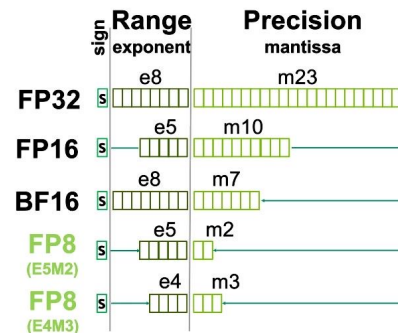
<https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/>

144 cores on the full GH100  
60MB L2 cache

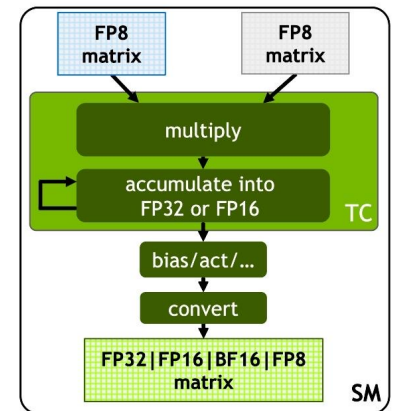
# NVIDIA H100 Core



48 TFLOPS Single Precision\*  
 24 TFLOPS Double Precision\*  
 800 TFLOPS (FP16, Tensor Cores)\*



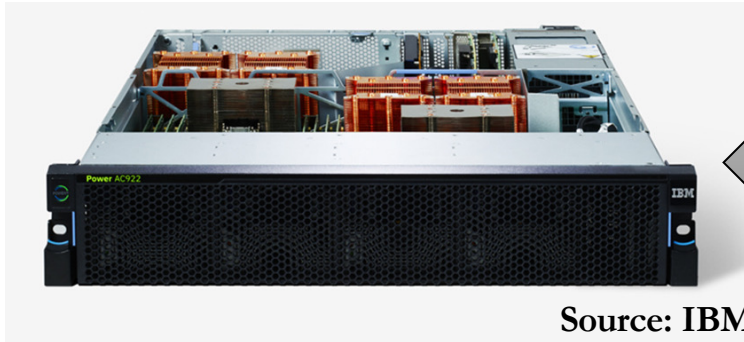
Allocate 1 bit to either  
 range or precision



Support for multiple accumulator  
 and output types

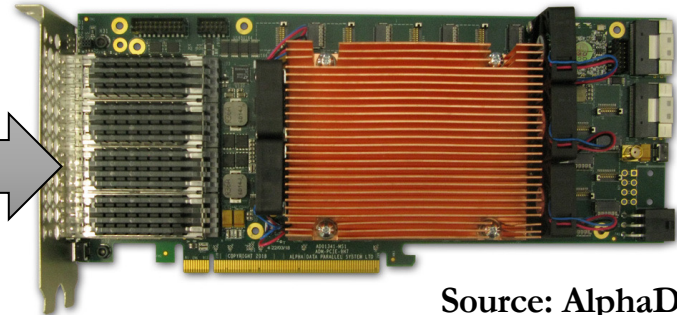


# Heterogeneous System: CPU+FPGA



Source: IBM

**POWER9 AC922**



Source: AlphaData

**HBM-based AD9H7 board**

We evaluate two POWER9+FPGA systems:

**1. HBM-based board AD9H7**

Xilinx Virtex Ultrascale+™ XCVU37P-2

**2. DDR4-based board AD9V3**

Xilinx Virtex Ultrascale+™ XCVU3P-2

# Accelerating Climate Modeling

---

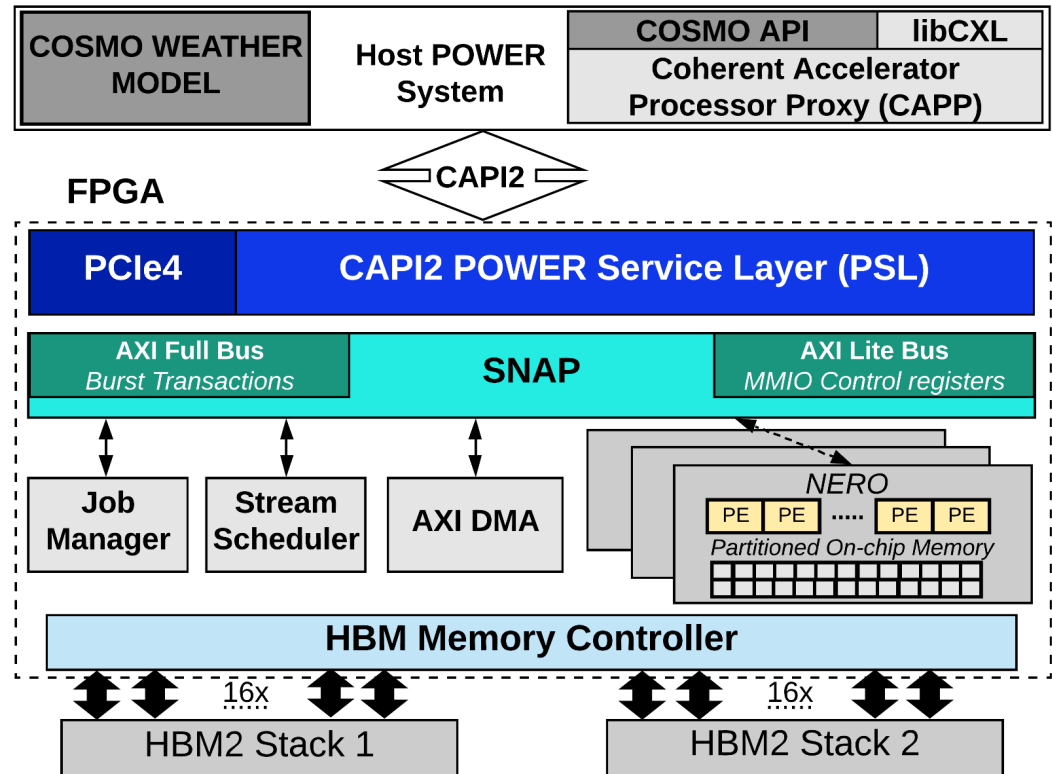
- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,  
**"NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"**  
*Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (FPL)*, Gothenburg, Sweden, September 2020.  
[[Slides \(pptx\)](#)] [[pdf](#)]  
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]  
[[Talk Video](#) (23 minutes)]  
***Nominated for the Stamatis Vassiliadis Memorial Award.***

## NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh<sup>a,b,c</sup>    Dionysios Diamantopoulos<sup>c</sup>    Christoph Hagleitner<sup>c</sup>    Juan Gómez-Luna<sup>b</sup>  
Sander Stuijk<sup>a</sup>    Onur Mutlu<sup>b</sup>    Henk Corporaal<sup>a</sup>  
<sup>a</sup>Eindhoven University of Technology    <sup>b</sup>ETH Zürich    <sup>c</sup>IBM Research Europe, Zurich

# NERO Application Framework

- NERO communicates to Host over **CAPI2** (Coherent Accelerator Processor Interface)
- **COSMO API** handles offloading jobs to NERO
- **SNAP** (Storage, Network, and Analytics Programming) allows for seamless integration of the COSMO API



<https://github.com/open-power/snap>

# FPGA-based Processing Near Memory

---

- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu, ["FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"](#) *IEEE Micro* (**IEEE MICRO**), 2021.

## FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh<sup>◇</sup> Mohammed Alser<sup>◇</sup> Damla Senol Cali<sup>✕</sup>

Dionysios Diamantopoulos<sup>▽</sup> Juan Gómez-Luna<sup>◇</sup>

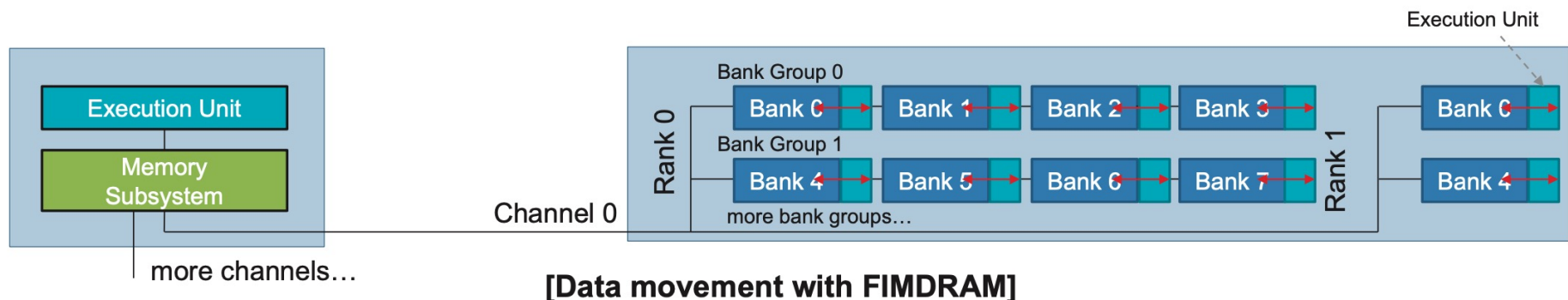
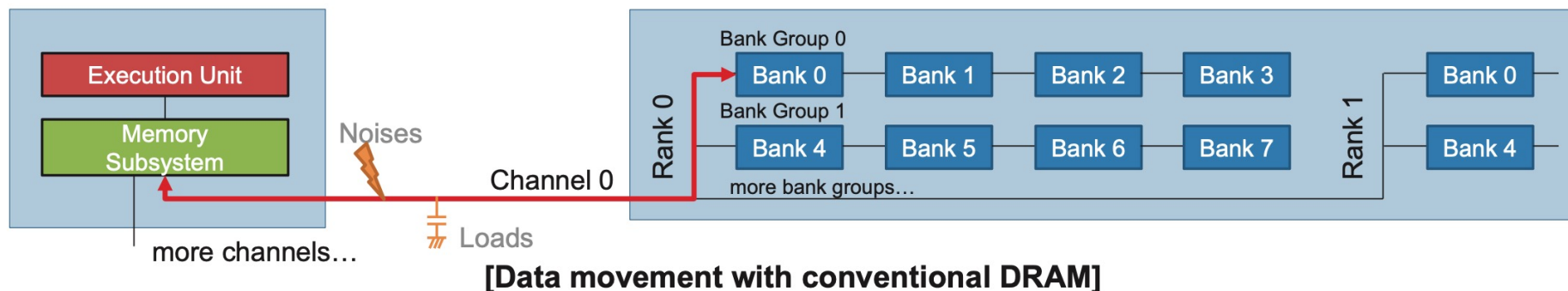
Henk Corporaal<sup>\*</sup> Onur Mutlu<sup>◇✕</sup>

<sup>◇</sup>*ETH Zürich*    <sup>✕</sup>*Carnegie Mellon University*

<sup>\*</sup>*Eindhoven University of Technology*    <sup>▽</sup>*IBM Research Europe*

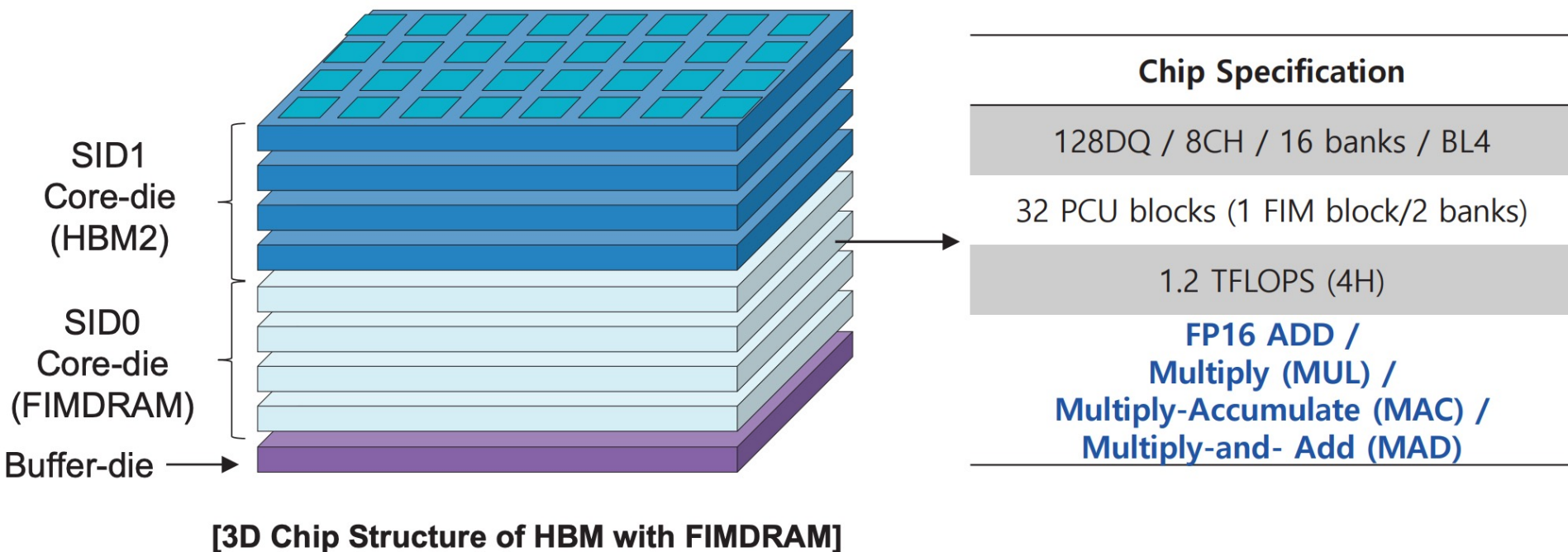
# FIMDRAM: Exploiting Bank Parallelism

- HBM bandwidth is not enough for many ML workloads
  - BLAS-1 and BLAS-2 are typically memory bound



# FIMDRAM: Chip Structure

## ■ FIMDRAM based on HBM2

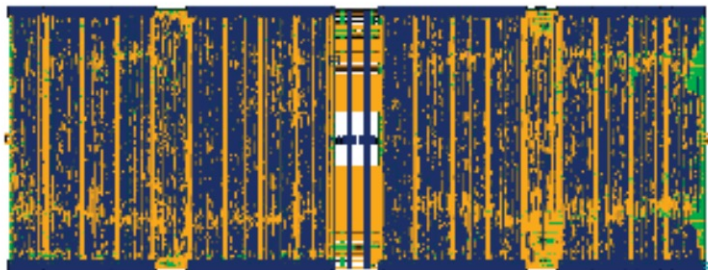




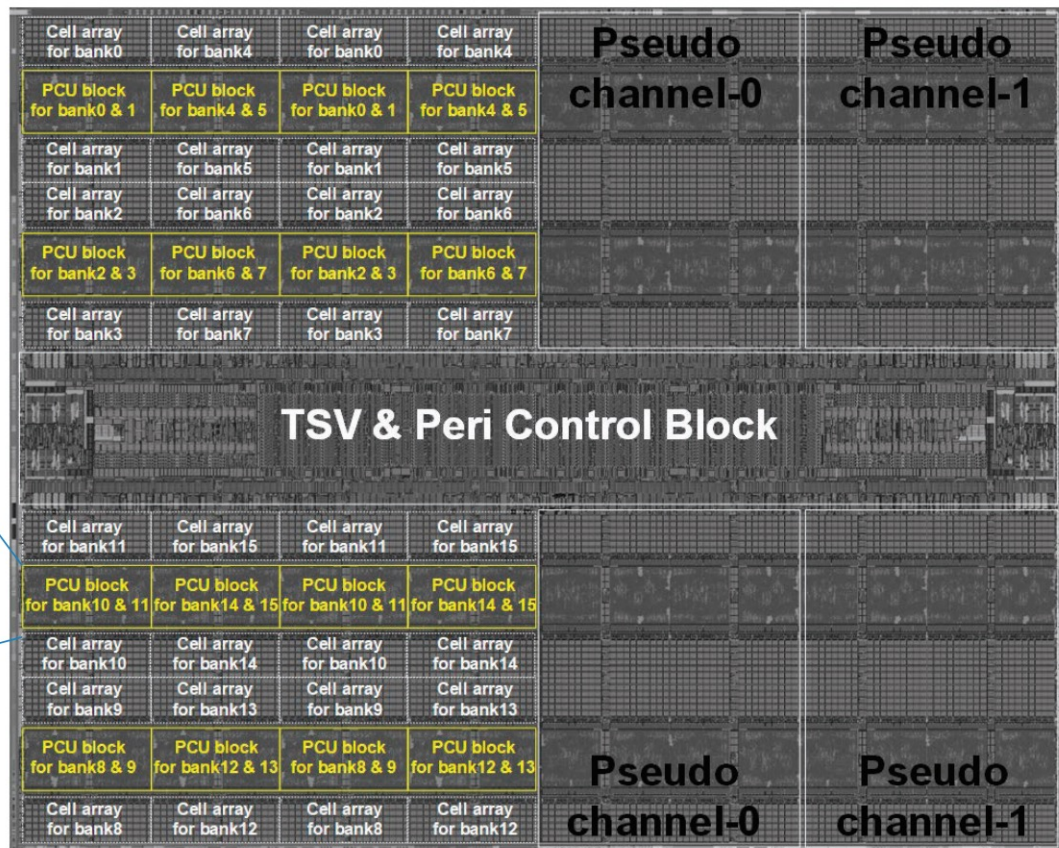
# FIMDRAM: Chip Implementation

## Chip Implementation

- Mixed design methodology to implement FIMDRAM
  - Full-custom + Digital RTL

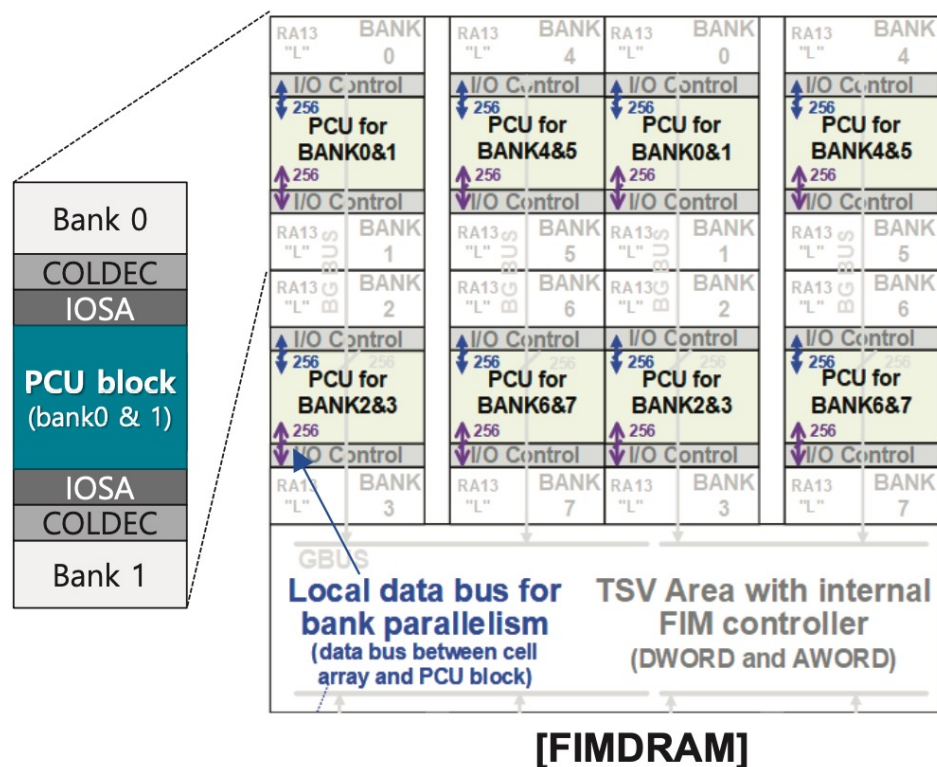
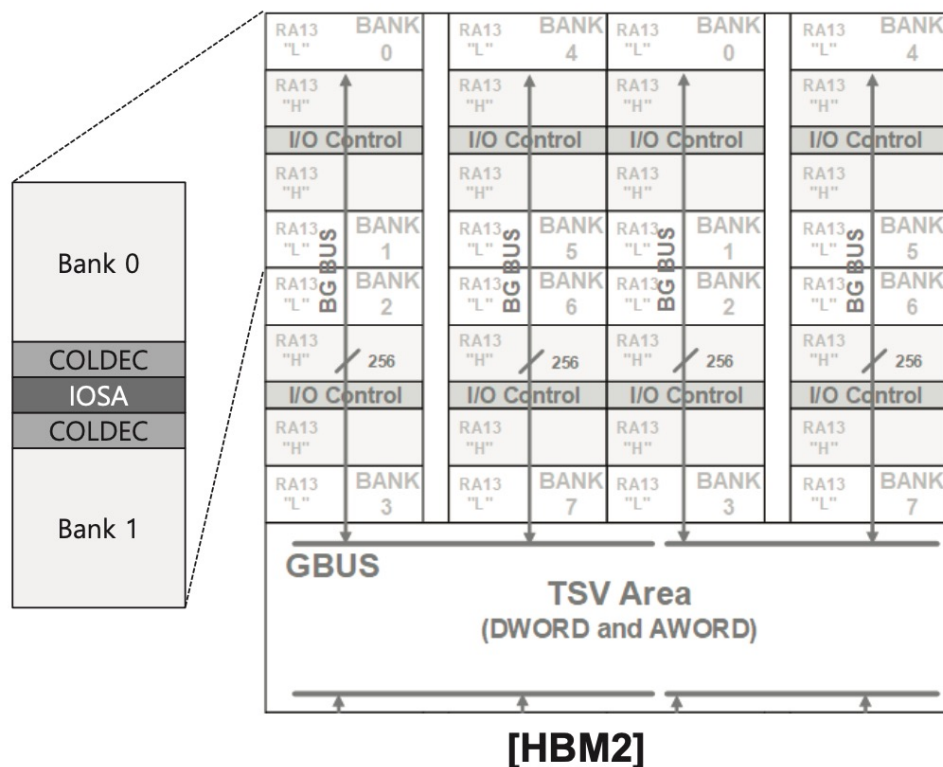


[Digital RTL design for PCU block]



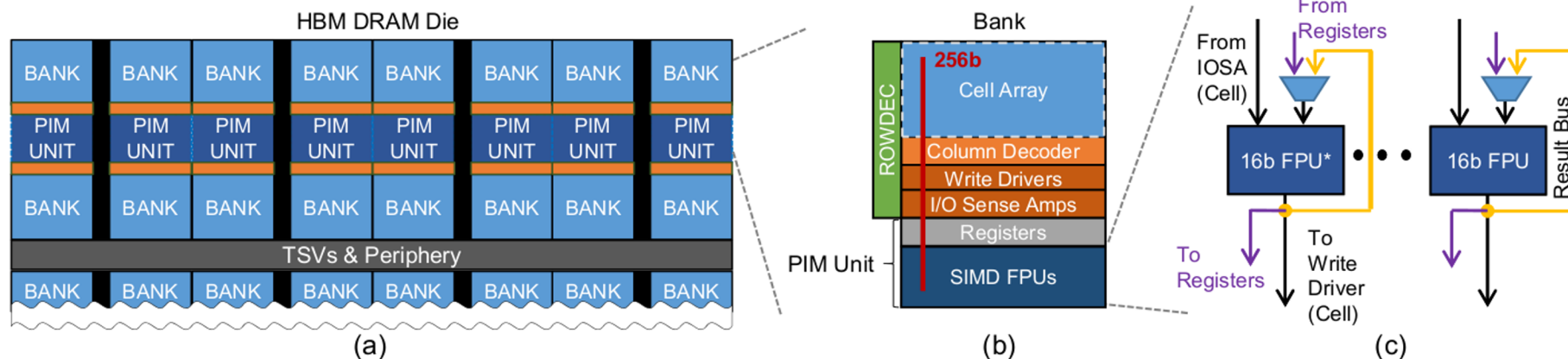
# FIMDRAM: System Organization (I)

## ■ HBM2 vs. FIMDRAM



# FIMDRAM: System Organization (II)

- Design goals:
  - ❑ 1. Support DRAM and PIM-DRAM mode for versatility
  - ❑ 2. Minimize the engineering cost of redesigning DRAM banks and sub-arrays
- Thus, PIM unit at I/O boundary of bank
  - ❑ 1 PIM unit for each 2 banks
  - ❑ 16 16-bit SIMD floating-point units (FPUs) per PIM unit



# SIMD Processing and GPUs

# Flynn's Taxonomy of Computers

---

- Mike Flynn, “[Very High-Speed Computing Systems](#),” Proc. of IEEE, 1966
- **SISD**: Single instruction operates on single data element
- **SIMD**: Single instruction operates on multiple data elements
  - Array processor
  - Vector processor
- **MISD**: Multiple instructions operate on single data element
  - Closest form: systolic array processor, streaming processor
- **MIMD**: Multiple instructions operate on multiple data elements (multiple instruction streams)
  - Multiprocessor
  - Multithreaded processor

# Data Parallelism

---

- Concurrency arises from performing the **same operation on different pieces of data**
  - Single instruction multiple data (SIMD)
  - E.g., dot product of two vectors
- Contrast with data flow
  - Concurrency arises from executing different operations in parallel (in a data driven manner)
- Contrast with thread (“control”) parallelism
  - Concurrency arises from executing different threads of control in parallel
- SIMD exploits operation-level parallelism on different data
  - Same operation concurrently applied to different pieces of data
  - A form of ILP where instruction happens to be the same across data

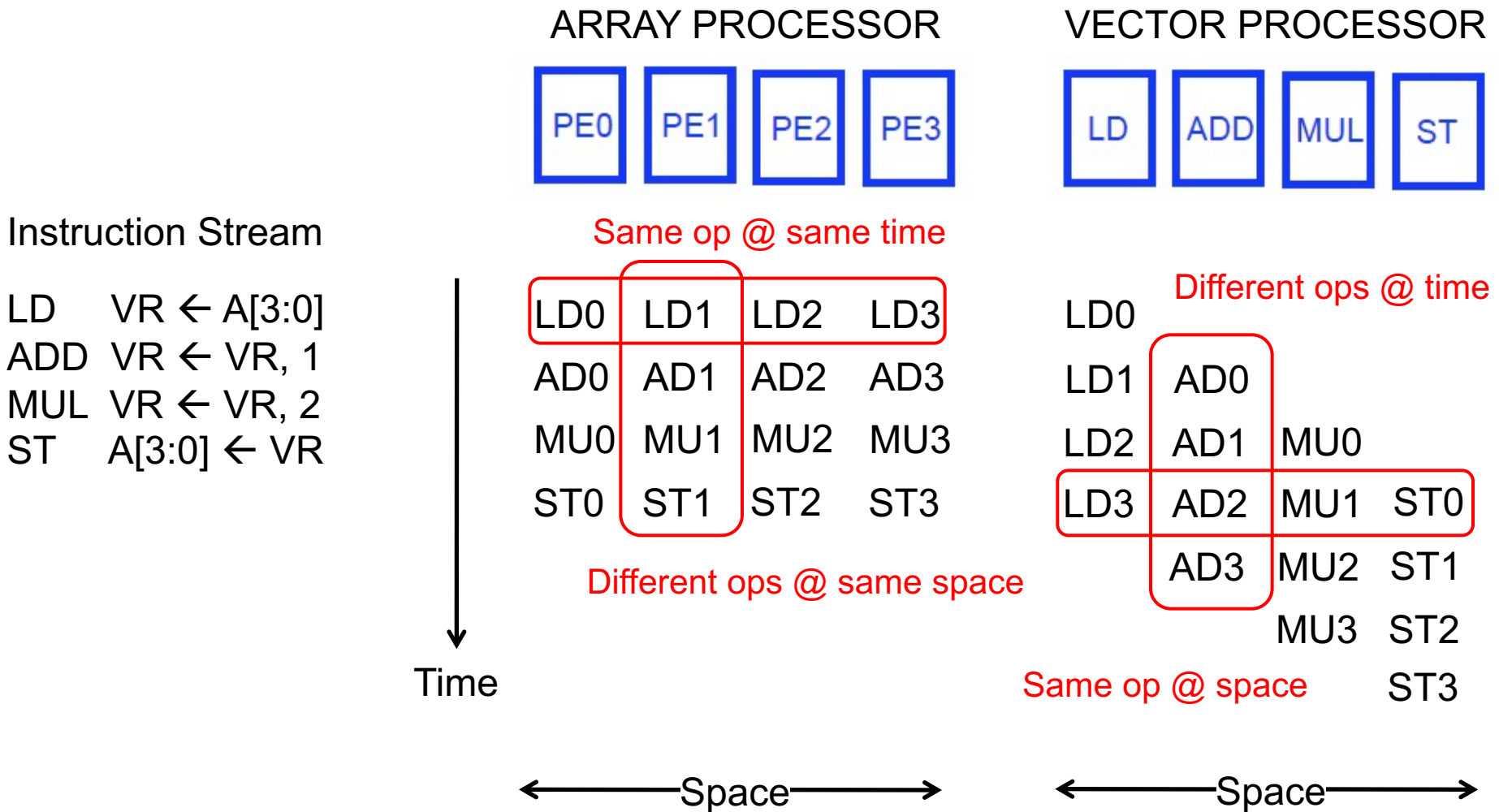


# SIMD Processing

---

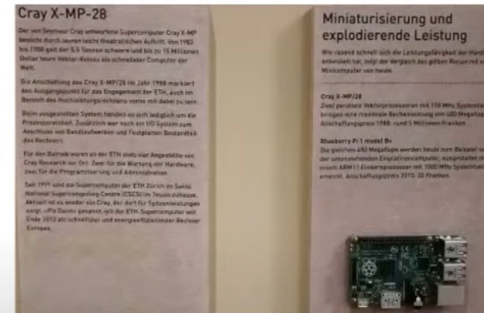
- Single instruction operates on multiple data elements
  - In time or in space
- Multiple processing elements (PEs), i.e., execution units
- Time-space duality
  - **Array processor**: Instruction operates on multiple data elements at the **same time** using **different spaces (PEs)**
  - **Vector processor**: Instruction operates on multiple data elements in **consecutive time steps** using the **same space (PE)**

# Array vs. Vector Processors



# Lecture on SIMD Processing

## CRAY X-MP-28 @ ETH (CAB, E Floor)



DEPARTMENT OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING (D-ITET)

Digital Design & Comp. Arch. - Lecture 20: SIMD Processing (Vector and Array Processors) (Spring'21)

2,677 views • Streamed live on May 14, 2021

69 1 SHARE SAVE ...



**Onur Mutlu Lectures**  
19.2K subscribers

SUBSCRIBED



# A GPU is a SIMD (SIMT) Machine

---

- Except it is **not** programmed using SIMD instructions
- It is **programmed using threads** (SPMD programming model)
  - Each thread executes the same code but operates a different piece of data
  - Each thread has its own context (i.e., can be treated/restarted/executed independently)
- A set of threads executing the same instruction are dynamically grouped into a **warp (wavefront)** by the hardware
  - A warp is essentially a **SIMD operation formed by hardware!**

# NVIDIA H100 Block Diagram



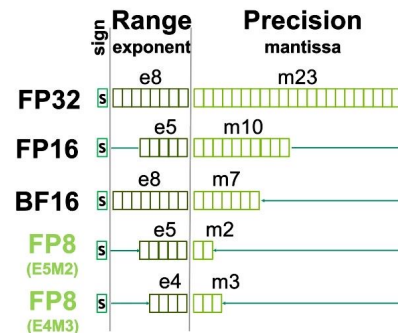
<https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/>

144 cores on the full GH100  
60MB L2 cache

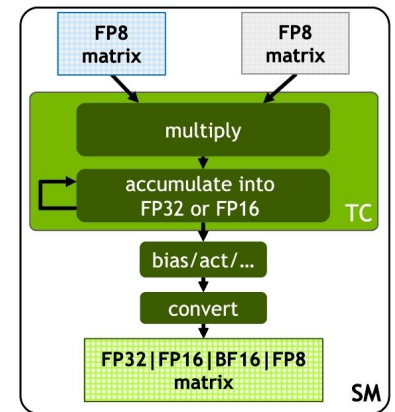
# NVIDIA H100 Core



48 TFLOPS Single Precision\*  
 24 TFLOPS Double Precision\*  
 800 TFLOPS (FP16, Tensor Cores)\*



Allocate 1 bit to either  
 range or precision

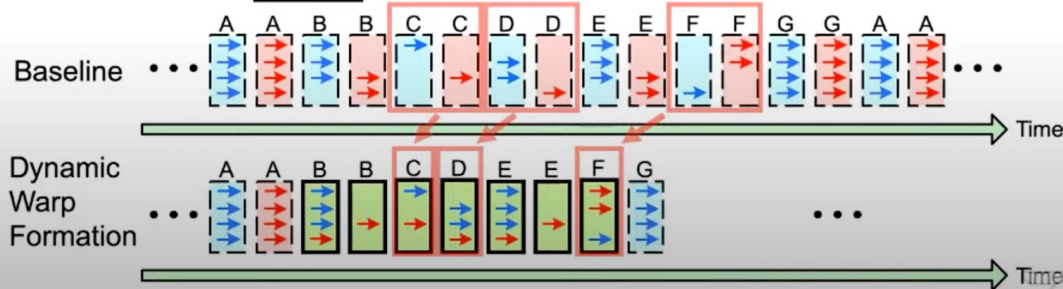
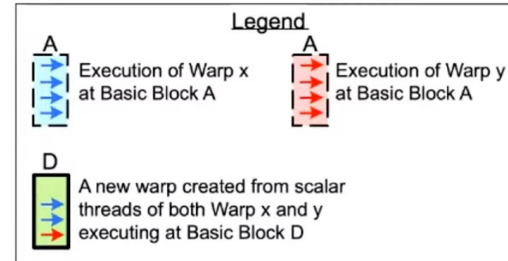
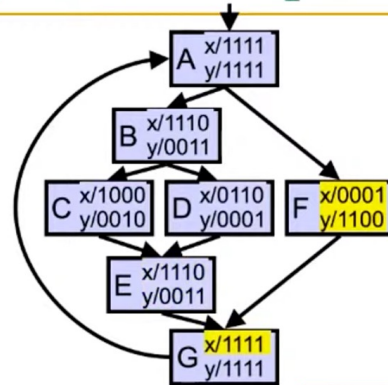


Support for multiple accumulator  
 and output types



# Lecture on Graphics Processing Units

## Dynamic Warp Formation Example



DEPARTMENT OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING (D-ITET)

Digital Design & Comp. Arch. - Lecture 21: Graphics Processing Units (GPUs) (ETH Zürich, Spring '21)

3,536 views • Streamed live on May 20, 2021

97 3 SHARE + SAVE ...



Onur Mutlu Lectures

19.5K subscribers

Digital Design and Computer Architecture, ETH Zürich, Spring 2021 (

<https://safari.ethz.ch/digitaltechnik...>)

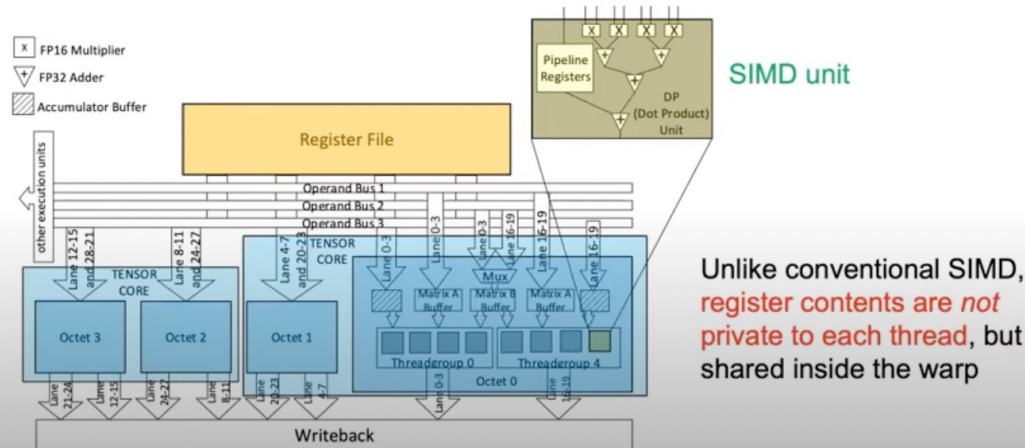
SUBSCRIBED



# Lecture on SIMD Processing and GPUs

## Tensor Core Microarchitecture (Volta)

- Each warp utilizes **two tensor cores**
- Each tensor core contains **two "octets"**
  - **16 SIMD units per tensor core** (8 per octet)
  - 4x4 matrix-multiply and accumulate each cycle per tensor core



~~Proposed\* tensor core microarchitecture~~

HetSys Course: Lecture 2: SIMD Processing and GPUs (Spring 2022)

380 views • Premiered Mar 22, 2022

9 DISLIKE SHARE CLIP SAVE ...



Onur Mutlu Lectures

23.6K subscribers

Project & Seminar, ETH Zürich, Spring 2022

Hands-on Acceleration on Heterogeneous Computing Systems (

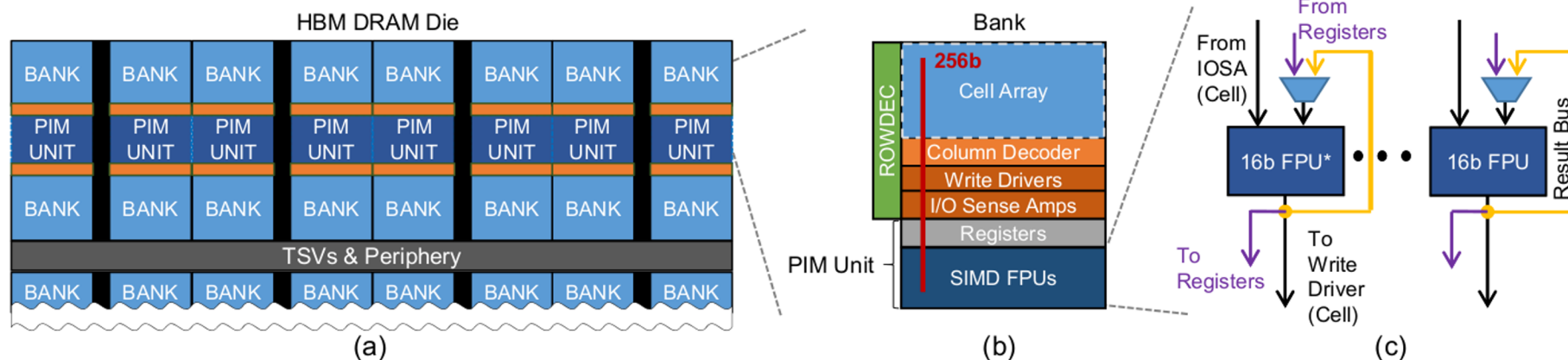
[https://safari.ethz.ch/projects\\_and\\_s...](https://safari.ethz.ch/projects_and_s...))

SUBSCRIBED



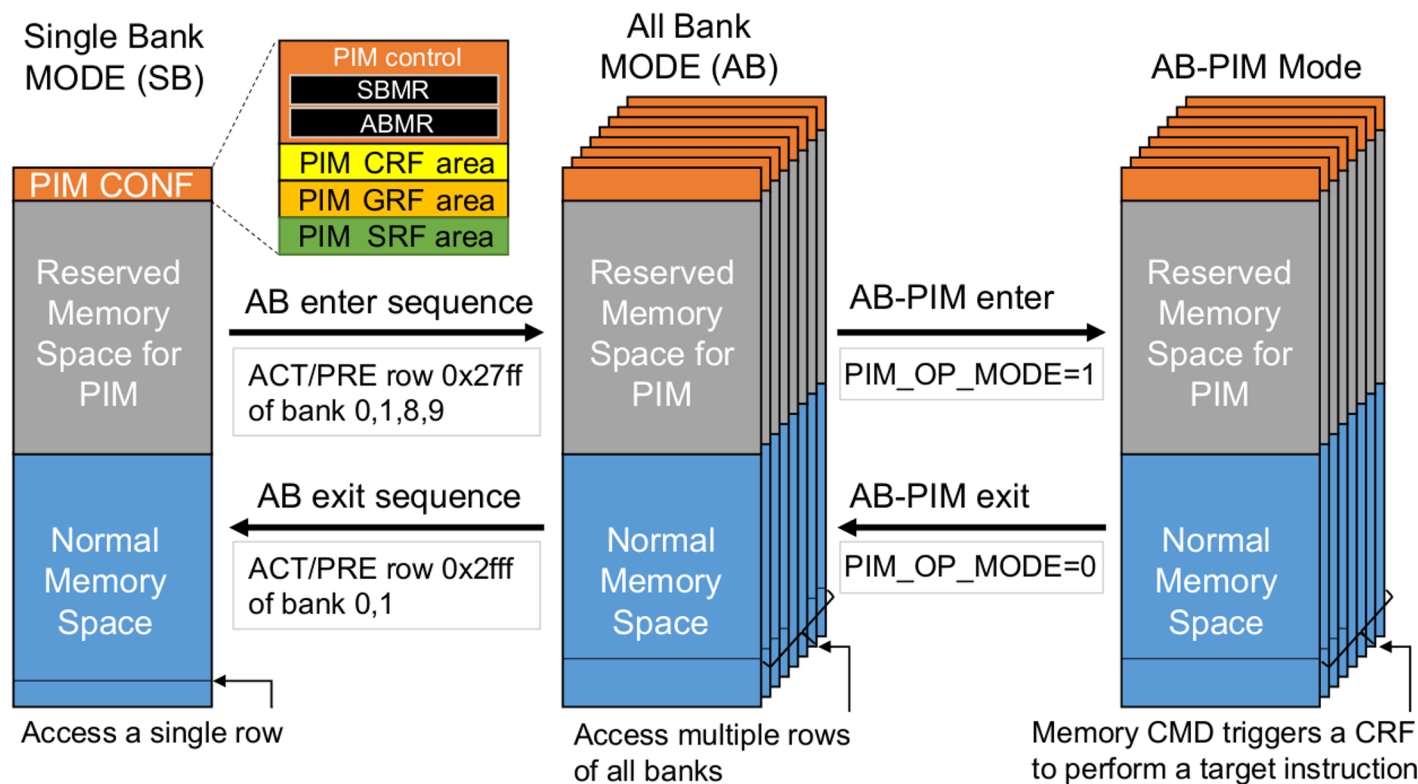
# FIMDRAM: System Organization (III)

- PIM units respond to standard DRAM column commands (RD or WR)
  - Compliant with **unmodified JEDEC controllers**
- They execute **one wide-SIMD operation** commanded by a **PIM instruction with deterministic latency in a lock-step manner**
- A PIM unit can get **16 16-bit operands** from IOSAs, a register, and/or the result bus



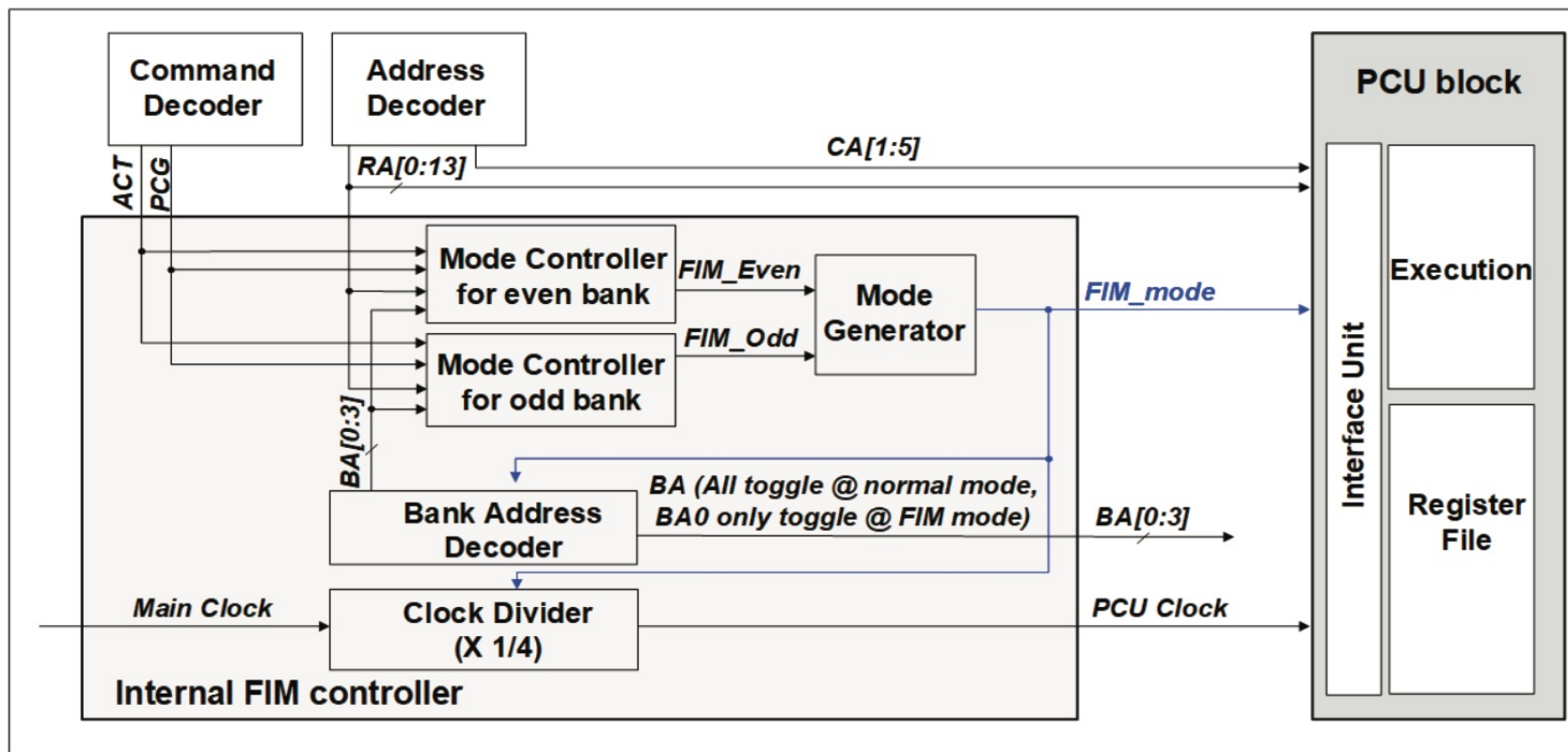
# FIMDRAM: Bank-level Parallelism

- Unlike standard DRAM devices, **all banks can be accessed concurrently for 8x higher bandwidth** (with 16 pCHs)
- In **AB-PIM mode**, a memory command triggers a PIM instruction in the CRF



# FIMDRAM: Internal FIM Controller

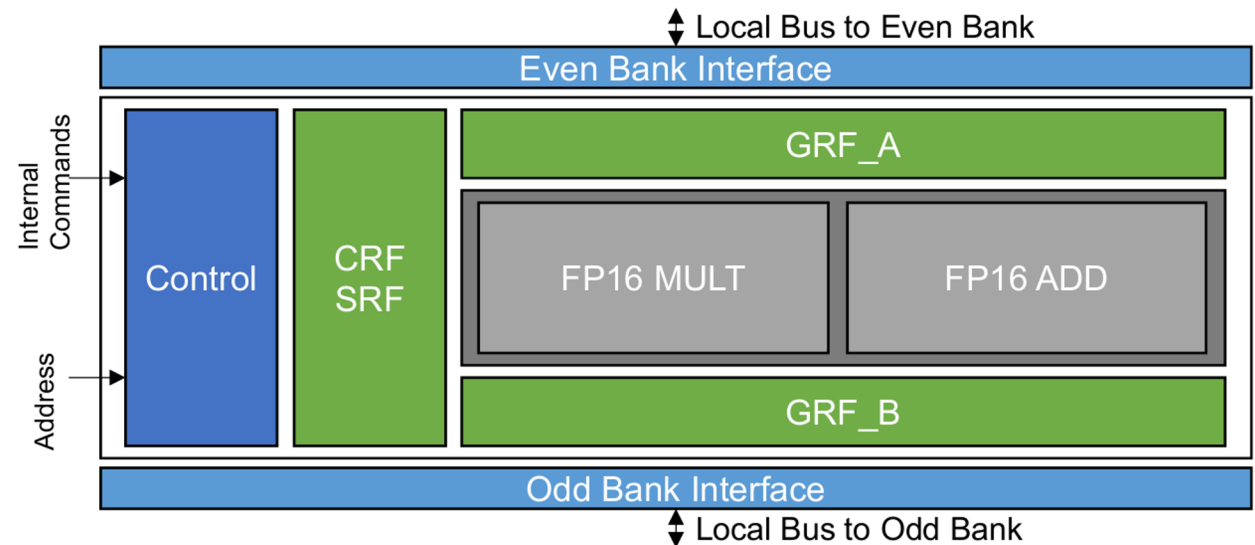
- The internal FIM controller controls FIM mode **without any modification of the host processor hardware**



[Block diagram of control circuit for FIM operation]

# FIMDRAM: Programmable Computing Unit (I)

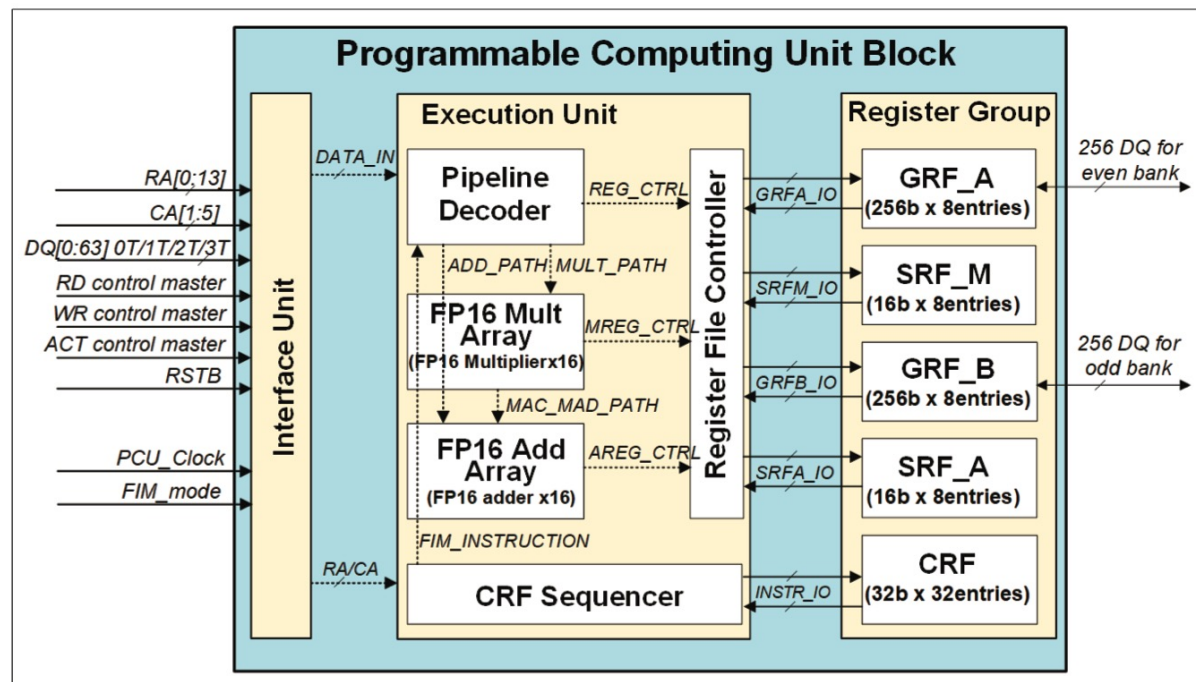
- Control: Instruction sequence manager
- Pipeline of 5 stages
  - ❑ 1. Fetch/decode
  - ❑ 2. Load 256-bit data from even or odd bank (optional)
  - ❑ 3. MUL
  - ❑ 4. ADD
  - ❑ 5. Writeback to GRF





# FIMDRAM: Programmable Computing Unit (II)

- Interface unit to control data flow
- Execution unit
- Register group
  - ❑ **CRF** (command): Instruction buffer
  - ❑ **GRF** (general): Weights and accumulation
  - ❑ **SRF** (source): Constants for MAC



[Block diagram of PCU in FIMDRAM]

# FIMDRAM: Instruction Set Architecture (I)

## ■ 9 RISC-style 32-bit instructions

[Available instruction list for FIM operation]

Type	CMD	Description
Floating Point	ADD	FP16 addition
	MUL	FP16 multiplication
	MAC	FP16 multiply-accumulate
	MAD	FP16 multiply and add
Data Path	MOVE	Load or store data
	FILL	Copy data from bank to GRFs
Control Path	NOP	Do nothing
	JUMP	Jump instruction
	EXIT	Exit instruction

# FIMDRAM: Instruction Set Architecture (II)

## ■ Combinations depend on operand sources

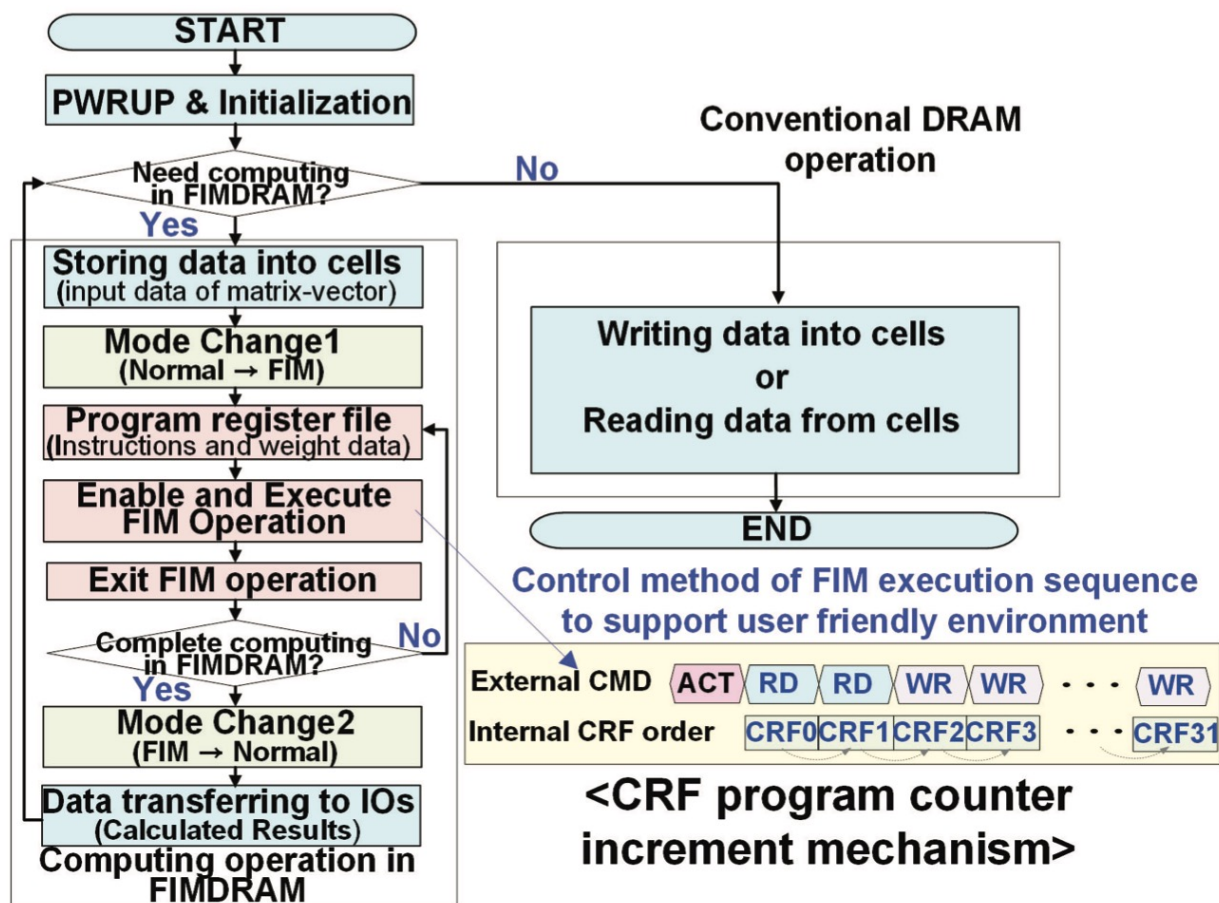
Op. Type	Operand (SRC0)	Operand (SRC1)	Result (DST)	# of Combinations
MUL	GRF, BANK	GRF, BANK, SRF_M	GRF	32
ADD	GRF, BANK, SRF_A	GRF, BANK, SRF_A	GRF	40
MAC	GRF, BANK	GRF, BANK, SRF_M	GRF_B	14
MAD	GRF, BANK	GRF, BANK, SRF_M SRF_A (for SRC2)	GRF	28
MOV (ReLU)	GRF, BANK		GRF	24

## ■ Instruction formats

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Control	OPCODE				U									IMM0									IMM1									
Data	OPCODE				DST				SRC0				U								R	U	DST #		U	SRC0 #		U	SRC1 #			
ALU	OPCODE				DST				SRC0				SRC1			SRC2			A	U			U	DST #		U	SRC0 #		U	SRC1 #		

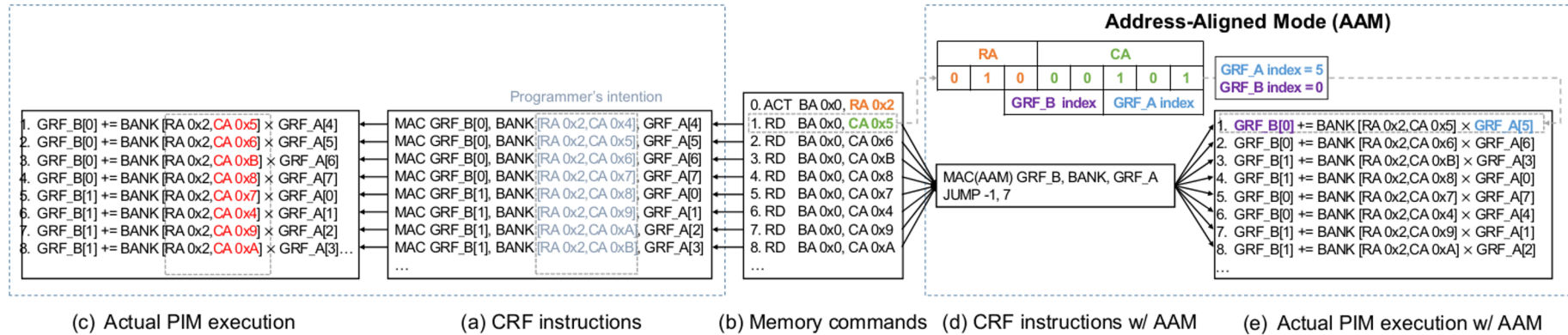
# FIMDRAM: Operation Flow

- Operation sequence for matrix vector computing
  - Input and output data are accessible to the host in conventional DRAM operation



# FIMDRAM: Instruction Ordering

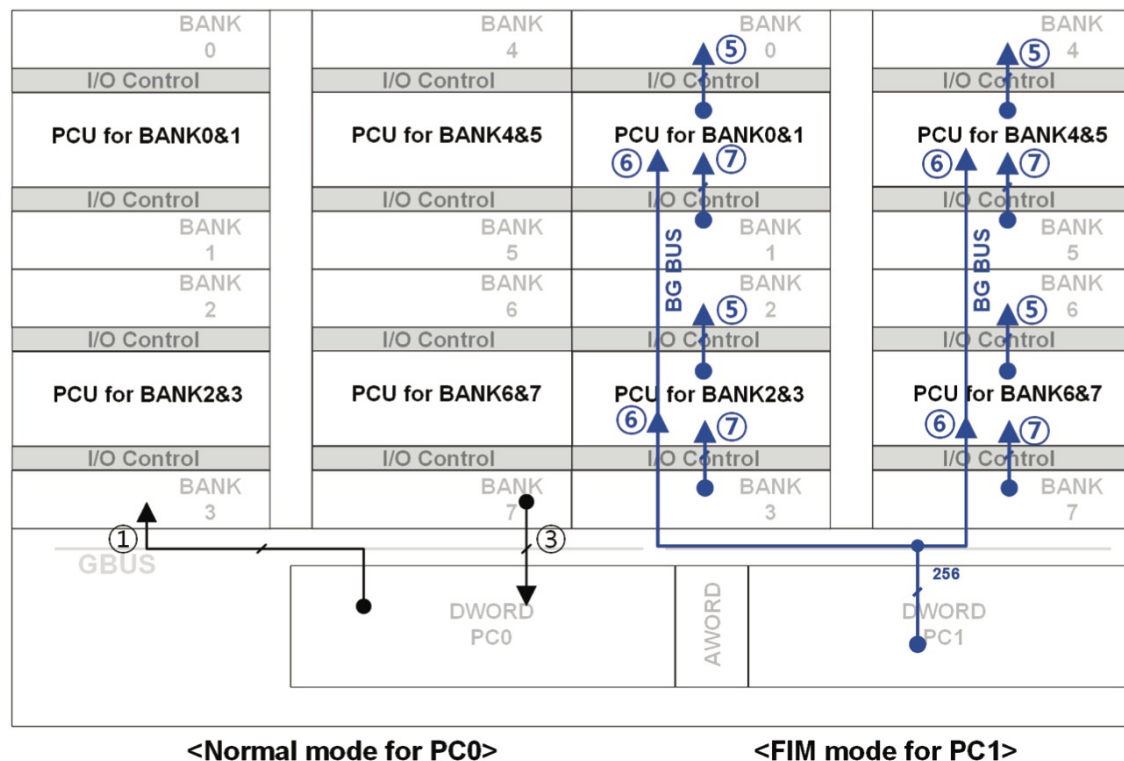
- One challenge is that DRAM commands may be re-ordered, and using fences is costly performance-wise
- Solution: **Address Aligned Mode (AAM)**
  - 8 MAC operations with 2 PIM instructions



# FIMDRAM: Data Flow

- Data flow controlled by operation mode and bit RA13

Index	Mode	CMD	RA13	Data Movement
①	Normal	WR	L	Data write to cell array
②		WR	H	Not available
③		RD	L	Data read to cell array
④		RD	H	Not available
⑤	FIM	WR	L	Data movement from PCU block to cell array
⑥		WR	H	Data write to PCU register
⑦		RD	L	Data movement from cell array to PCU block
⑧		RD	H	Not available





# FIMDRAM: Key Feature Summary

## ■ Comparison table

	[3]	[7]	UPMEM PIM [8]	FIMDRAM (this work)
Type of DRAM	HBM2	LPDDR4	DDR4	HBM2
Process	20 nm	20 nm	2x nm	20 nm
Memory density	8GB/cube (Buffer-die + 8H 8Gb core-die)	8GB/chip (8H 8Gb mono die)	8GB/DIMM	6GB/cube (Buffer-die + 4H 4Gb core-die with PCU + 4H 8Gb core-die )
Data rate	2.4Gbps	3.2Gbps	2.4Gbps	2.4Gbps
Bandwidth	307GB/s per cube	25.6GB/s per chip	19.2GB/s per DIMM	307GB/s per cube
# of CH	8 per cube	1 per chip	16 per DIMM	8 per cube
# of processing unit	No	2048 per chip (256 per die)	128 per DIMM (8 per chip)	128 per cube (32 per core-die)
Processing operation speed	-	250Mhz @simulation	500MHZ @ Measurement	300MHZ @ Measurement
Peak throughput	-	0.5 TOPS per chip (250MHz x 256 x 8byte)	0.5 TOPS per DIMM (500MHz x 128 x 8byte)	1.2 TFLOPS per cube (300MHz x 128 x 32byte)
Operation Precision	-	INT8	INT8	FP16

TFLOPS : Tera Floating Point Operations Per Second

[3] K. Sohn, et al., ISSCC 2016, [7] H. Shin, et al., IEEE TCADICS 2018, [8] F. Devaux, IEEE Hot Chips Symp. 2019

# Function-in-Memory DRAM (ISSCC 2021)

---

## ISSCC 2021 / SESSION 25 / DRAM / 25.4

---

### **25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications**

Young-Cheon Kwon<sup>1</sup>, Suk Han Lee<sup>1</sup>, Jaehoon Lee<sup>1</sup>, Sang-Hyuk Kwon<sup>1</sup>,  
Je Min Ryu<sup>1</sup>, Jong-Pil Son<sup>1</sup>, Seongil O<sup>1</sup>, Hak-Soo Yu<sup>1</sup>, Haesuk Lee<sup>1</sup>,  
Soo Young Kim<sup>1</sup>, Youngmin Cho<sup>1</sup>, Jin Guk Kim<sup>1</sup>, Jongyoon Choi<sup>1</sup>,  
Hyun-Sung Shin<sup>1</sup>, Jin Kim<sup>1</sup>, BengSeng Phuah<sup>1</sup>, HyoungMin Kim<sup>1</sup>,  
Myeong Jun Song<sup>1</sup>, Ahn Choi<sup>1</sup>, Daeho Kim<sup>1</sup>, SooYoung Kim<sup>1</sup>, Eun-Bong Kim<sup>1</sup>,  
David Wang<sup>2</sup>, Shinhaeng Kang<sup>1</sup>, Yuhwan Ro<sup>3</sup>, Seungwoo Seo<sup>3</sup>, JoonHo Song<sup>3</sup>,  
Jaeyoun Youn<sup>1</sup>, Kyomin Sohn<sup>1</sup>, Nam Sung Kim<sup>1</sup>

<sup>1</sup>Samsung Electronics, Hwaseong, Korea

<sup>2</sup>Samsung Electronics, San Jose, CA

<sup>3</sup>Samsung Electronics, Suwon, Korea

## Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology

Industrial Product

Sukhan Lee<sup>§1</sup>, Shin-haeng Kang<sup>§1</sup>, Jaehoon Lee<sup>1</sup>, Hyeonsu Kim<sup>2</sup>, Eojin Lee<sup>1</sup>, Seungwoo Seo<sup>2</sup>,  
Hosang Yoon<sup>2</sup>, Seungwon Lee<sup>2</sup>, Kyoungwan Lim<sup>1</sup>, Hyunsung Shin<sup>1</sup>, Jinhyun Kim<sup>1</sup>,  
Seongil O<sup>1</sup>, Anand Iyer<sup>3</sup>, David Wang<sup>3</sup>, Kyomin Sohn<sup>1</sup> and Nam Sung Kim<sup>§1</sup>

<sup>1</sup>Memory Business Division, Samsung Electronics

<sup>2</sup>Samsung Advanced Institute of Technology, Samsung Electronics

<sup>3</sup>Device Solutions America, Samsung Electronics

## **Aquabolt-XL: Samsung HBM2-PIM with in-memory processing for ML accelerators and beyond**

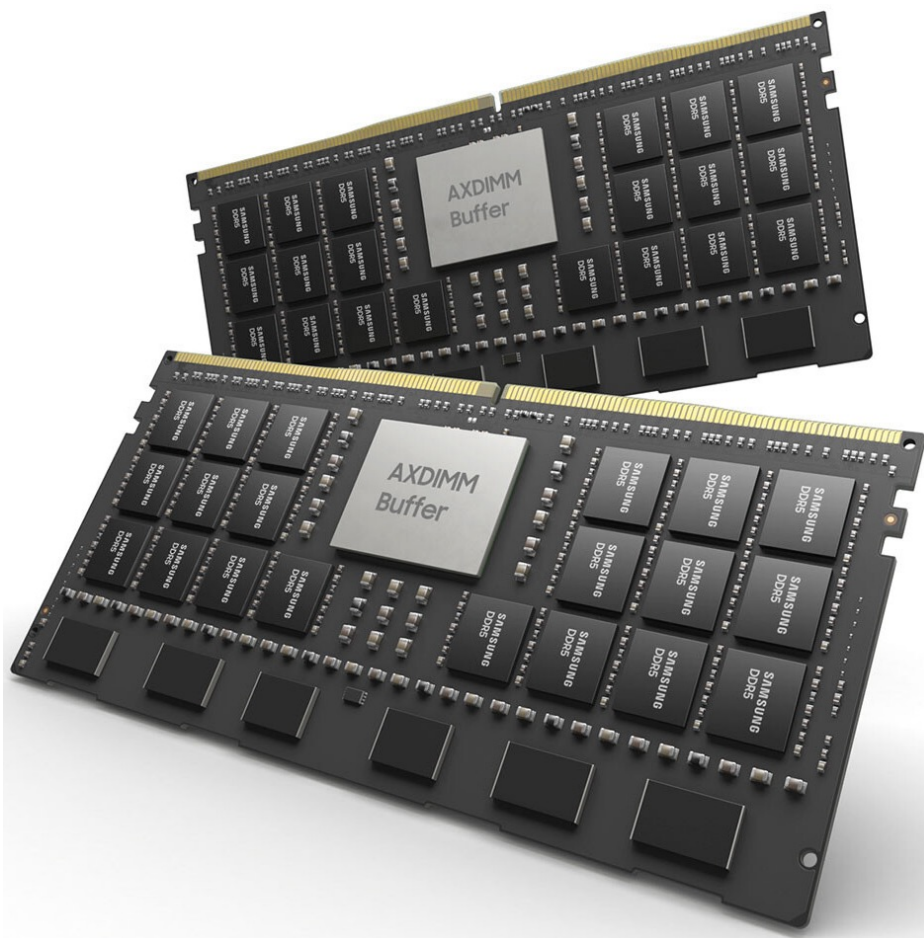
Jin Hyun Kim, Shin-haeng Kang, Sukhan Lee, Hyeonsu Kim, Woongjae Song, Yuhwan Ro, Seungwon Lee, David Wang, Hyunsung Shin, Bengseng Phuah, Jihyun Choi, Jinin So, YeonGon Cho, JoonHo Song, Jangseok Choi, Jeonghyeon Cho, Kyomin Sohn, Youngsoo Sohn, Kwangil Park, and Nam Sung Kim

Samsung Electronics

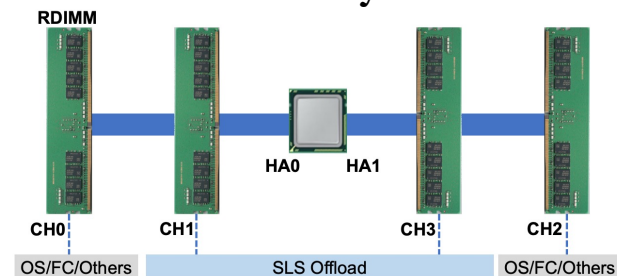


# Samsung AxDIMM (2021)

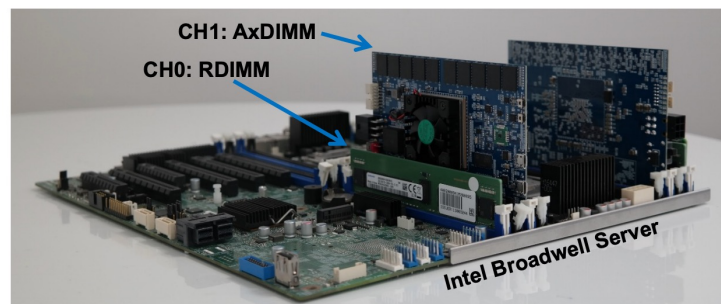
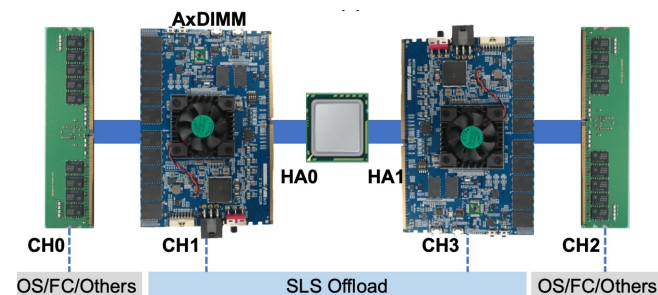
- DIMM-based PIM
  - DLRM recommendation system



Baseline System



AxDIMM System



# Upcoming Lectures

---

- More real-world PIM architectures
- Programming PIM systems
- Workload characterization for PIM suitability



# P&S Processing-in-Memory

Real-World Processing-in-Memory Architectures:  
Samsung HBM-PIM Architecture

Dr. Juan Gómez Luna

Prof. Onur Mutlu

ETH Zürich

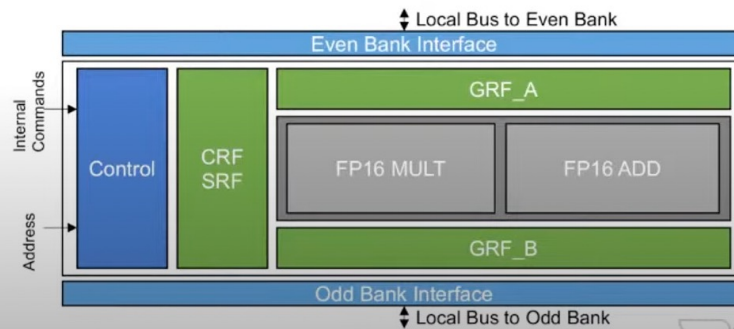
Fall 2022

8 November 2022

# Another Lecture on HBM-PIM

## FIMDRAM: Programmable Computing Unit

- Control: Instruction sequence manager
- Pipeline of 5 stages
  - ❑ 1. Fetch/decode
  - ❑ 2. Load 256-bit data from even or odd bank (optional)
  - ❑ 3. MUL
  - ❑ 4. ADD
  - ❑ 5. Writeback to GRF



Livestream - P&S Exploring the Processing-in-Memory Paradigm for Future Computing Systems (Fall 2021)

Processing in Memory Course: Meeting 4: Real-world PIM architectures III - Fall'21



Onur Mutlu Lectures

28.8K subscribers

Subscribed



21



Share

Clip

Save



780 views Streamed 1 year ago

Project & Seminar, ETH Zürich, Fall 2021

Exploring the Processing-in-Memory Paradigm for Future Computing Systems ([https://safari.ethz.ch/projects\\_and\\_s...](https://safari.ethz.ch/projects_and_s...))

...more